(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/90892 A1

(51) International Patent Classification[7]: G06F 9/445, 1/00

(21) International Application Number: PCT/US01/15720

(22) International Filing Date: 14 May 2001 (14.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/580,931 25 May 2000 (25.05.2000) US

(71) Applicant *(for all designated States except US)*: **EVERDREAM, INC.** [US/US]; 6591 Dumbarton Circle, Fremont, CA 94555 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MCCALEB, Jed** [US/US]; 1663 Fulton, San Francisco, CA 94117 (US). **RIVE, Russell** [CA/US]; 2433 Greer Road, Palo Alto, CA 94303 (US).

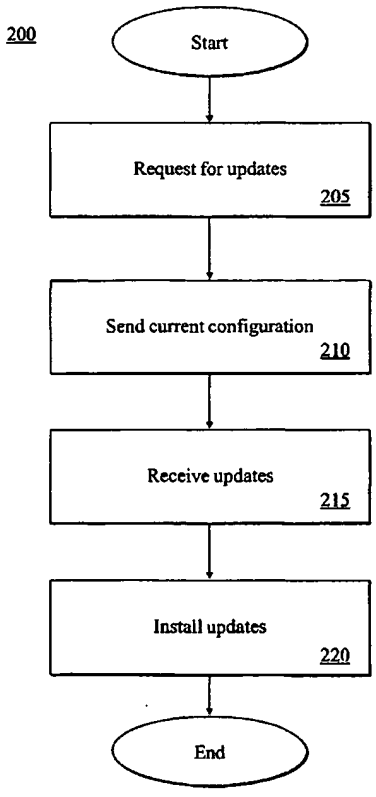
(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: INTELLIGENT PATCH CHECKER



(57) Abstract: A method for remotely updating software in a plurality of computer systems is disclosed. In one embodiment, a request for an upgrade is sent to a server system connected in a network. The upgrade is for a software application installed in a client connected in the network. The request is sent from the client system. The request comprises a unique identification associated with the client system. The unique identification is recognized by the server system as belonging to the client system. At least one instruction is received from the server system in response to the request for the upgrade. The at least one instruction directs the client system to collect application information about the software application installed on the client system. The application information about the software application is sent to the server system. The server system performs a comparison between the application information about the software application and the most-updated upgrade package for the software application. The most-updated upgrade package for the software application is received by the client system automatically when the comparison indicates that the most-updated upgrade package has not been installed on the client system.

WO 01/90892 A1

patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTELLIGENT PATCH CHECKER

FIELD OF THE INVENTION

The present invention relates generally to field of remote support for computer systems. More specifically, the present invention is directed to a method and an apparatus for updating software in a plurality of computer systems.

BACKGROUND

Personal computers have become an important part of the information age. The use of the personal computers has expanded beyond the traditional university campus and large office environments. Today, many small businesses and residences have at least one personal computer running a wide range of applications sold by many different software vendors.

As the applications become easier to use, the personal computers are no longer considered the tool for only the technical users. The user community has expanded and the personal computers are being viewed more as the tools to run the applications. Most users are interested in dealing with the applications and usually have no clue when something goes wrong with their personal computers. When the user is unable to use the application on the user's personal computer, the usual action is to take the personal computer to a local personal computer repair shop.

Since there are many different brands of personal computers such as, for example, IBM, Compaq, Gateway, Dell, etc., it is usually the case that each personal computer from a different brand may have a different set up. For example, the IBM personal computer may use a different video adapter from the Dell personal computer, among others. As such, to have a problem corrected, the user usually has to bring the personal computer into the repair shop so that the technician can isolate the problem.

One of the most common problems of application failure is incompatibility. The incompatibility may be related to the hardware or to the other applications in the same personal computer system. For example, the user may have installed a new application that is incompatible with the existing application when running together. The user may have installed a new hardware adapter that is incompatible with the existing application without installing a necessary update. Often the identification of the incompatibility occurs at a most unfortunate time such as, for example, prior to the user having an opportunity to save the work in progress. This experience is frustrating, time consuming and can be costly for the user.

SUMMARY OF THE INVENTION

A method for remotely updating software in a plurality of computer systems is disclosed. In one embodiment, a request for an upgrade is sent to a server system connected in a network. The upgrade is for a software application installed in a client system connected in the network. The request is sent from the client system. The request comprises a unique identification associated with the client system. The unique identification is recognized by the server system as belonging to the client system. At least one instruction is received from the server system in response to the request for the upgrade. The server system has knowledge of the software application installed on the client system. The at least one instruction directs the client system to collect application information about the software application installed on the client system. The server system has no knowledge whether most-updated upgrade packages available for the software application have been installed on the client system. The application information about the software application is sent to the server system. The server system performs a comparison between the application information about the software application and the most-updated upgrade package for the software application. The most-updated upgrade

package for the software application is stored in a part database. The most-updated upgrade package for the software application is received by the client system automatically when the comparison indicates that the most-updated upgrade package has not been installed on the client system.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example in the following drawings in which like references indicate similar elements. The following drawings disclose various embodiments of the present invention for purposes of illustration only and are not intended to limit the scope of the invention.

Figure 1 is a network diagram illustrating one embodiment of components connected in a network that can be used with the method of the present invention.

Figure 2 is a flow diagram illustrating one embodiment of an update process.

Figure 3 is another flow diagram illustrating one embodiment of the update process.

Figure 4 is an exemplary tool bar that can be used with one method of the

present invention.

Figure 5 is an exemplary diagram illustrating a relationship between a server connection point, a customer data base and a part data base.

Figure 6 is an exemplary diagram illustrating a communication protocol between a client system and the server through the Internet network

Figure 7 illustrates one embodiment of a computer-readable medium containing various sets of instructions, code sequences, configuration information, and other data used by a computer or other processing device.

DETAILED DESCRIPTION

A method and apparatus for remotely updating software in a plurality of computer systems is disclosed. In the following description, for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention.

Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within

the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method operations. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

In one embodiment, the method disclosed in the present invention allows for better remote support of users of client systems in the network. A server provides update information to multiple client systems connected in a network. When necessary, the updates are retrieved from a central depository, sent to the appropriate client systems and automatically update the applications. In one embodiment, the client systems are IBM-compatible personal computers running in the Window environment such as, for example, Windows 98, Windows 2000, etc. The server and the client systems are connected in a network such as, for example, the Internet. By keeping the client systems updated, remote support can be efficiently performed to

minimize the down time of the client systems. Each client system comprises of multiple installed software packages. The software packages may have been previously installed on the client system prior to delivery to a user. The software may include, for example, application software, device drivers, etc.

Figure 1 illustrates an exemplary embodiment of the update network. A server 105 maintains a client database 125 to keep track of the client systems 110, 115. For example, whenever a client system 110 or 115 communicates with the server 105, the server 105 already knows about the installed software on that client system 110, 115. The server 105 also maintains a part database 120 containing software patches and software updates to help keeping the client systems 110 and 115 up to date. The client database 125 allows the server 105 to know about the configuration of the client systems 110 and 115. The client database 125 and the part database 120 may be in the same database server or in separate database servers connected in the network 130. Alternatively, the client database 125 and the part database 120 may be in the same system as the server 105. In one embodiment, the server 105 serves as a central point for receiving update requests from the client systems 110 and 115 and for retrieving information from the databases 125 and 120 to satisfy the update requests.

Figure 2 is a flow diagram 200 illustrating one embodiment of an update method. At block 205, an update request is generated by the client system 110, 115 and sent to the server 105. The update is performed on a periodic basis, such as, for example, every 24 hours. Alternatively, the update may be performed at any time by the user sending an update request to the server 105 on the network. The server 105 knows each client system 110, 115 by a unique identification associated with the client system 110, 115.

In one embodiment, the server 105 accesses a client database 125 containing information about the client system 110, 115. The client database 125 may include information, such as, for example, installed software packages on the client system 110, 115, the operating system installed on the client system 110, 115, etc. However, what the server 105 may not know is whether

these installed software packages are up to date. For example, the user of the client system 110, 115 may have changed the configuration parameters of the software packages, or the user may not have requested for an update for an extended length of time due to the client system 110, 115 not being connected to the network 130.

In one embodiment, the client system 110, 115 may need to do a self-check and send its current software configuration to the server 105. A self-check may be done by the server 105 directing the client system 110, 115 specifically what to check for and the information to be collected from the client system 110,115. This information is the sent to the server 105, as shown in block 210. Based on this information, the server 105 checks its part database 120 and determines the updates that the client system 110, 115 needs. The updates are sent from the server 105 to the client system 110, 115, as shown in block 215. The updates may be sent with instructions from the server 105 that tells the client system 110, 115what to do to have the updates installed, as shown in block 220.

Figure 3 is another flow diagram illustrating one embodiment of an update method 300. In one embodiment, a utility program executed by the client system 110, 115 communicates with the server 105 for information to check on the client system 110, 115. The execution of this utility program may be initiated by the user or it may be automatic. The utility program is herein referred to as a patch checker. The patch checker initiates the request to have the applications verified for any necessary updates. The request is sent to the server 105 along with the unique identification number of the client system 110, 115. The server 105 uses the client identification number to check against the client database 125 for authentication. In one embodiment, the database contains configuration information about the client system 110, 115. The server 105 retrieves the configuration information for the client system 110, 115, generates a script file and sends the script file to the client system 110, 115, as shown in block 305. In one embodiment, the script file contains commands that tell the client system 110, 115 the functions to perform. For example, the

commands may direct the client system 110, 115 to perform self-check functions. The self-check functions may have the following parameters:

- 'v: filename' get the file's version
- 'm: filename' get the file's modified date
- 'd: driveletter' get amount of free disk space
- 'r: keyname' get the value of the specified registry key
- 's: filename' get the size of the file.

In one embodiment, the commands are executed by the client system 110, 115 to collect information pertinent to the applications currently installed on the client system 110, 115.

The script file may contain a list of parts that the server 105 thinks the client system 110, 115 has and that the server 105 wants the client system 110, 115 to check. The parts may be the names of the applications and the server 105 may want the client system 110, 115 to collect the current version information about the applications. In one embodiment, in order to keep the information in the client database 125 accurate, the user may not want to alter the configuration of the applications that are to be supported remotely by the server 105. Keeping the client system 110, 115 and the information in the client database 125 synchronized may help making the update process by the server 105 more efficient.

In block 310, using the script information sent by the server 105, the patch checker parses the server's commands to check the software parts on the client system 110, 115. The appropriate information about these software parts is collected. In one embodiment, the version of each software part is collected and sent to the server 105, as shown in block 315. The server 105 uses the information collected from the client system 110, 115 and compares it with a part database 120. For example, the server 105 may check the version number collected from the client system 110, 115 with the version of the same software part in the part database 120. In one embodiment, the server 105 may want to get the most updated version distributed to the client system 110, 115.

When the version information collected from the client system 110, 115 is not at the same level with the version of the same software part in the part database 120, the most updated version is retrieved from the part database 120. When the version information from the client system 110, 115 is already up to date, there is nothing to download. In block 320, the patch checker asks the server 105 for the files associated with the updated versions of the software to download. The files are downloaded from the server 105 to the client system 110, 115 in block 325. In one embodiment, each download file is associated with an uniform resource locator (URL). The server 105 replies to the update request by sending the patch URL.

There may be one or more download files for each software to be updated, and there may be more than one software that needs to be updated, the server 105 may send several down load files to the client system 110, 115. The download files may be stored in a predefined directory such as, for example, the download directory. Each download file is processed individually, as shown in block 330. In one embodiment, the download files are received from the server 105 in a compressed format, such as, the zip format, and need to be uncompressed or expanded, as shown in block 335. Each download file is expanded into an executable program and multiple related data files. One of the data files is a text file or an instruction file containing a set of instructions or commands that can be parsed by the executable program to perform the update process, as shown in block 340. For example, the instruction may be one of the following commands:

- Delete a file
- ShellExecute
- ShellExecute with wait
- Registry Change
- Add message to a tool bar
- Kill a particular process
- Ask for reboot
- Force reboot

- Ask user to install now or later
- Ask user to close all programs

When all of the download files have been expanded and copied into the appropriate directories, the update process is completed. At that time, the user may be given an option of rebooting the client system 110, 115 to activate the updated version. Alternatively, the user may continue working with the currently installed version and reboot the client system 110, 115 at a later time.

Figure 4 illustrates an exemplary tool bar that can be used with the present invention. In one embodiment, the tool bar is a list of dynamic link libraries (DLL) and is always running. Additional functions can be added to the tool bar by adding DLL files. For example, the patch checker can be added to the tool bar 400 by adding a patcher.dll to the tool bar 400, and a patch checker icon can be displayed. By selecting the patch checker icon, the user can initiate the update process at any time. In one embodiment, the tool bar 400 is also used to display update related messages to the user.

Figure 5 is an exemplary diagram illustrating a relationship between a connection point, a customer data base and a part data base. In one embodiment, the server provides a connection point 505 that connects to a customer database 510. The customer database 510 maintains the state of every client system in the network. The state includes information concerning relevant hardware and software as currently installed on the client system. This information includes, for example, the versions of the installed software applications, the versions of the installed hardware drivers, etc. Additionally, the connection point 505 is also connected to a part database 515. The part database 515 may contain the different versions of the application software, the DLLs, the hardware drivers, and any other software modules that may be installed on the client system. The server uses the part database 515 to keep the client system up to date. For example, when the client system is identified to have a hardware driver that is not current, the most up-to-date hardware driver is retrieved from the part database 515.

In one embodiment, a client part database is maintained in the client system. The client part database contains the versions of the software that are installed on the client system. As additional software is installed on the client system, the client part database is updated accordingly. In one embodiment, when the server wants to know the versions of the software installed on the client system, the patch checker retrieves the version information from the client part database.

Figure 6 is an exemplary diagram illustrating a communication protocol between a client system 600 and a server 650 through the Internet network. In one embodiment, the client system 600 has a message queue 605 to store messages that certain application 610, such as, for example, the patch checker (patcher.dll), wants to send to the server 650. The user selects the patch checker icon on the tool bar 400 displayed by the tool bar program 620 (dreambar.exe) to execute the patch checker 610. The message queue 605 is processed periodically, such as, for example, every thirty minutes, while the user is connected to the Internet 690. When the user is not connected to the Internet 690, the messages from the patch checker (applications) 610 are stored in the message queue 605. In one embodiment, the patch checker 610 connects to the server 650 through a message handler 615 (ConPoint.dll). The message handler 615 handles the messages generated by the patch checker 610 including, for example, the request for an update. The message handler 615 sends the message to the server 650. In another embodiment, the message queue 605 is implemented as a text file located in a message queue directory.

In one embodiment, the server 650 is implemented with multiple java servlets. A master servlet 655 (AnnexServlet) is used to route all the messages received from the client systems 600 to the other servlets 665, 670 on the server 650. Each of the servlets 660, 665, 670 handles different type of messages. In one embodiment, as each servlet starts up, the servlet tells the master servlet which type of messages the servlet 660, 665, 670 handles. The master servlet 655 may be used as the connection point on the server 650. Each of the servlets 660, 665, 670 may be used as a worker. For example, the serverlet 660 is the

patch worker handling the update messages from the patch checker 610. The patch worker 660 sends the script file to the patch checker 610. The script file is used by the patch checker 610 to check the client system 600. When the patch checker 610 requests for the download, the patch worker 660 accesses the part database 665 to retrieve the necessary software versions for the client system 600. It will be apparent to one skilled in the art that there may be other workers (servlets) on the server 650, such as, for example, a buildworker to add a new client system to the client database, a viewworker to view contents of the client database 680 and the part database 675, a dataworker to store data, and a messageworker to get the messages to be displayed on the tool bar 400.

In one embodiment, each client system 600 is associated with a unique identification number known to the server 650. As a new client system 600 is inserted into the network, the client database 680 is updated with the identification number of that new client system 600. Similarly, when the client system 600 is removed from the network, the client database 680 is updated accordingly. In one embodiment, the server 650 generates a report listing all the identification number of those client systems 600 that have not communicated with the server 650 for over a predetermined length of time. The report can then be used to investigate status of these client systems.

Figure 7 illustrates an embodiment of a computer-readable medium 700 containing various sets of instructions, code sequences, configuration information, and other data used by a computer or other processing device. The embodiment illustrated in **Figure 7** is suitable for use with the software update method described above. The various information stored on medium 700 is used to perform various data processing operations. Computer-readable medium 700 is also referred to as a processor-readable medium. Computer-readable medium 700 can be any type of magnetic, optical, or electrical storage medium including a diskette, magnetic tape, CD-ROM, memory device, or other storage medium.

Computer-readable medium 700 includes interface code 705 that controls the flow of information between various devices or components in the

computer system. Interface code 705 may control the transfer of information within a device (e.g., between the processor and a memory device), or between an input/output port and a storage device. Additionally, interface code 705 may control the transfer of information from one device to another or from one network component to another.

Computer-readable medium 700 also includes the patch checker program 710 that is used to request and receive software patches or updates from the server. Other codes stored on the computer-readable medium 700 may include the tool bar program 715 to display the patch checker icon, the message queue handler program 720 to receive the messages generated by the patch checker and send the messages to the server. The computer-readable medium 700 may also contain programs run on the server. These programs may include the patch worker 725 that communicates with the patch checker 710 from the server side, and the database access program 730 that allows the server to view the client database and the part database.

From the above description and drawings, it will be understood by those of ordinary skill in the art that the particular embodiments shown and described are for purposes of illustration only and are not intended to limit the scope of the invention. Those of ordinary skill in the art will recognize that the invention may be embodied in other specific forms without departing from its spirit or essential characteristics. References to details of particular embodiments are not intended to limit the scope of the claims.

CLAIMS

What is claimed is:

1. A method comprising:

sending a request for an upgrade to a server system connected in a network, the upgrade being for a plurality of software applications installed in a client system connected in the network, the request sent from the client system, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

receiving at least one instruction from the server system in response to the request for the upgrade, the server system having a knowledge of the software applications installed on the client system, the at least one instruction directing the client system to collect application information about the software applications installed on the client system, the server system having no knowledge whether most-updated upgrade packages available for the software applications have been installed on the client system;

sending the application information about the software applications to the server system, wherein the server system performs a comparison between the application information about the software applications and the most-updated upgrade packages for the software applications, wherein the most-updated upgrade packages for the software applications are stored in a part database; and

receiving the most-updated upgrade packages for the software applications at the client system automatically when the comparison indicates that the most-updated upgrade packages have not been installed on the client system.

2. The method of claim 1, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

3. The method of claim 2, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software applications installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

4. The method of claim 3, wherein the knowledge of the software applications installed on the client system comprises names of the software applications installed on the client system.

5. The method of claim 3, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

6. The method of claim 5, wherein the network is an Internet.

7. The method of claim 5, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

8. The method of claim 5, wherein the client database is in the server system.

9. The method of claim 1, wherein the application information about the software applications comprises version information of the software applications, and wherein the application information about the software applications is stored in a database in the client system.

10. The method of claim 1, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

11. The method of claim 10, wherein the predetermined time interval is 24 hours.

12. The method of claim 1, wherein the request is sent at any time by a user using the client system.

13. The method of claim 1, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

14. The method of claim 1, wherein receiving the most-updated upgrade packages for the software applications from the server system comprises receiving a plurality of download files.

15. The method of claim 14, wherein the plurality of download files are in a compressed format.

16. The method of claim 15, wherein each download file comprises an upgrade utility, a text file and a plurality of data files, wherein the text

file provides commands to the upgrade utility to install the plurality of data files in the client system.

17. The method of claim 16, wherein the commands comprise copy a file, delete a file, registry change, ask for reboot, force reboot, ask the user to install now or later, and ask the user to close all programs.

18. The method of claim 16, wherein each down load file is associated with a uniform resource locator (URL), and wherein each down load file is retrieved by accessing the URL.

19. A machine-readable medium providing instructions, which when executed by a set of one or more processors, cause said set of processors to perform the following:

sending a request for an upgrade to a server system connected in a network, the upgrade being for a plurality of software applications installed in a client system connected in the network, the request sent from the client system, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

receiving at least one instruction from the server system in response to the request for the upgrade, the server system having a knowledge of the software applications installed on the client system, the at least one instruction directing the client system to collect information about the software applications installed on the client system, the server system having no knowledge whether most-updated upgrade packages available for the software applications have been installed on the client system;

sending the information about the software applications to the server system, wherein the server system performs a comparison between

the information about the software applications and the most-updated upgrade packages for the software applications, wherein the most-updated upgrade packages for the software applications are stored in a part database; and

receiving the most-updated upgrade packages for the software applications to the client system automatically when the comparison indicates that the most-updated upgrade packages have not been installed on the client system.

20. The machine-readable medium of claim 19, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

21. The machine-readable medium of claim 20, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software applications installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

22. The machine-readable medium of claim 21, wherein the knowledge of the software applications installed on the client system comprises names of the software applications installed on the client system.

23. The machine-readable medium of claim 21, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

24. The machine-readable medium of claim 23, wherein the network is an Internet.

25. The machine-readable medium of claim 23, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

26. The machine-readable medium of claim 23, wherein the client database is in the server system.

27. The machine-readable medium of claim 19, wherein the information about the software applications comprises version information of the software applications, and wherein the information about the software applications is stored in a database in the client system.

28. The machine-readable medium of claim 19, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

29. The machine-readable medium of claim 28, wherein the predetermined time interval is 24 hours.

30. The machine-readable medium of claim 19, wherein the request is sent at any time by a user using the client system.

31. The machine-readable medium of claim 19, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

32. The machine-readable medium of claim 19, wherein receiving the most-updated upgrade packages for the software applications from the server system comprises receiving a plurality of download files.

33. The machine-readable medium of claim 32, wherein the plurality of download files are in a compressed format.

34. The machine-readable medium of claim 33, wherein each download file comprises an upgrade utility, a text file and a plurality of data files, wherein the text file provides commands to the upgrade utility to install the plurality of data files in the client system.

35. The machine-readable medium of claim 34, wherein the commands comprise copy a file, delete a file, registry change, ask for reboot, force reboot, ask the user to install now or later, and ask the user to close all programs.

36. The machine-readable medium of claim 34, wherein each down load file is associated with a uniform resource locator (URL), and wherein each down load file is retrieved by accessing the URL.

37. In an arrangement comprising at least one computer network, the network connecting at least one server computer to at least one client computer, a data processing system for providing software upgrades to the client computer, comprising:

means for generating a request for a software upgrade, the software upgrade being for an application on the client computer;

means for processing the request for the software upgrade comprising:

means for retrieving current information about the application on the client computer;

means for comparing the current information about the application on the client computer with information about an updated package for the application, the updated package stored in a part database accessible by the server computer; and

means for sending the updated package from the part database to the client computer when the current information about the application on the client computer and the information about the updated package are not the same.

38. The data processing system of claim 37 further comprising means for verifying a unique identification associated with the client system against a client database accessible by the server computer, the client database comprising the unique identification associated with the client computer, the client computer previously registered with the server computer.

39. The data processing system of claim 37, wherein the current information about the application on the client computer and the information about the updated package comprise version information.

40. The data processing system of claim 37, wherein the request for the software upgrade is generated automatically at a predetermined time interval.

41. In an arrangement comprising at least one computer network, the network connecting at least one server computer to at least one client computer, a data processing system for providing software upgrades to the client computer, comprising:

a first logic in the client computer to generate a request for a software upgrade, the software upgrade being for an application installed on the client computer; and

a second logic in the server computer to process the request for the software upgrade received from the first logic, the second logic comprising:

logic to extract current information about the application installed on the client computer;

logic to compare the current information about the application installed on the client computer with information about a most updated upgrade package for the application installed on the client computer, the most updated upgrade package stored in a first database; and

logic to send the most updated upgrade package for the application to the client computer when the current information about the application installed on the client computer does not match the information about the most updated upgrade package for the application.

42. The data processing system of claim 41 further comprising a third logic on the server computer to verify an identification associated with the client computer against a set of valid identifications stored in a client database, wherein the identification associated with the client computer is added to the set of valid identifications when the client computer is registered with the server computer.

43. The data processing system of claim 41, wherein the current information about the application installed on the client computer comprises version numbers of the applications.

44. The data processing system of claim 41, wherein the request for software upgrade is automatically generated by the client computer at a predetermined time interval.

45. The data processing system of claim 41, wherein the request for software upgrade is generated at any time by a user.

46. A method comprising:

receiving a request for an upgrade from a client system connected in a network, the upgrade being for a software application installed in a client system, the request received at a server system connected to the network, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

sending at least one instruction from the server system to the client system in response to the request for the upgrade, the server system having a knowledge of the software application installed on the client system, the at least one instruction directing the client system to collect information about the software application installed on the client system, the server system having no knowledge whether most-updated upgrade package available for the software application have been installed on the client system;

receiving the information about the software application from the client system, wherein the server system performs a comparison between the information about the software application and the most-updated upgrade package for the software application, wherein the most-

updated upgrade package for the software application is stored in a database; and

sending the most-updated upgrade package for the software application to the client system automatically when the comparison indicates that the most-updated upgrade package have not been installed on the client system.

47. The method of claim 46, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

48. The method of claim 47, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software application installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

49. The method of claim 48, wherein the knowledge of the software application installed on the client system comprises name of the software application installed on the client system.

50. The method of claim 48, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

51. The method of claim 50, wherein the network is an Internet.

52. The method of claim 50, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

53. The method of claim 50, wherein the client database is in the server system.

54. The method of claim 46, wherein the information about the software application comprises version information of the software application, and wherein the information about the software application is stored in a database in the client system.

55. The method of claim 46, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

56. The method of claim 46, wherein the request is sent at any time by a user using the client system.

57. The method of claim 46, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

58. The method of claim 46, wherein receiving the most-updated upgrade package for the software application from the server system comprises at lest one download file.

59. The method of claim 58, wherein the at least one download file is associated with a uniform resource locator (URL), and wherein the at least one down load file is retrieved by accessing the URL.

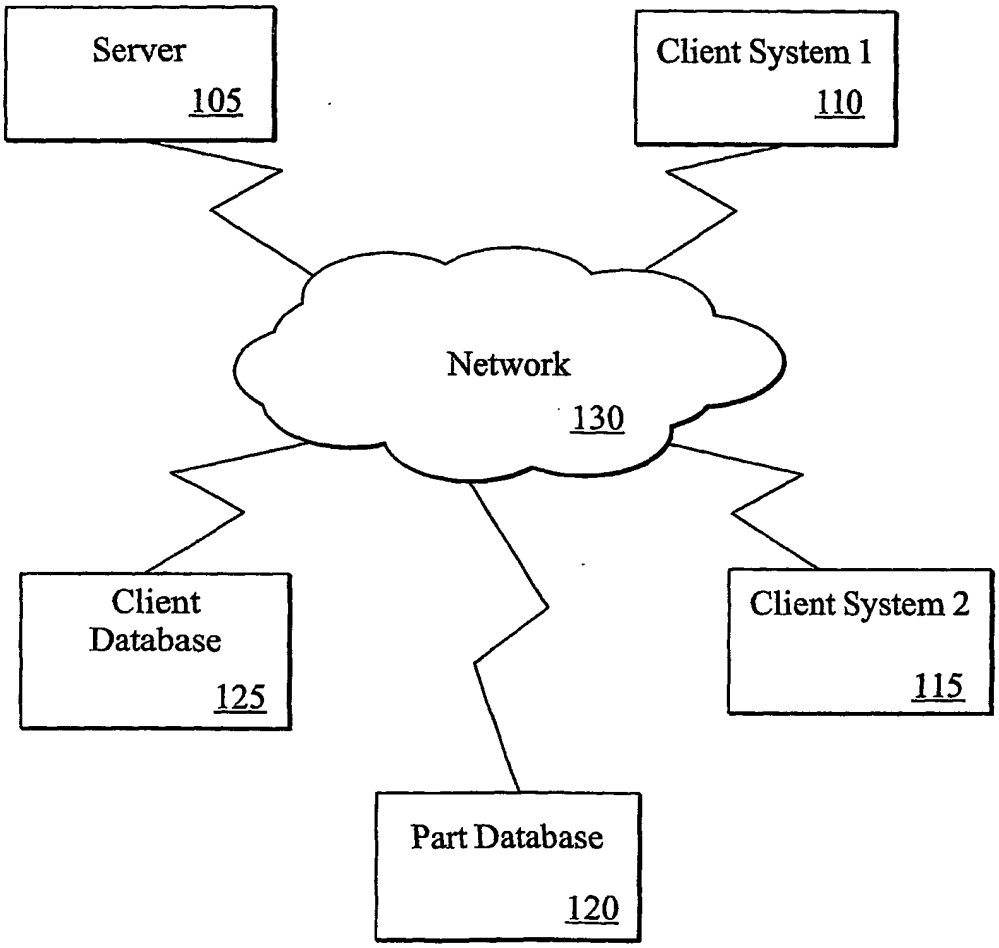

Server
105
Client System 1
110
Network
130
Client
Database
125
Client System 2
115
Part Database
120

Fig. 1

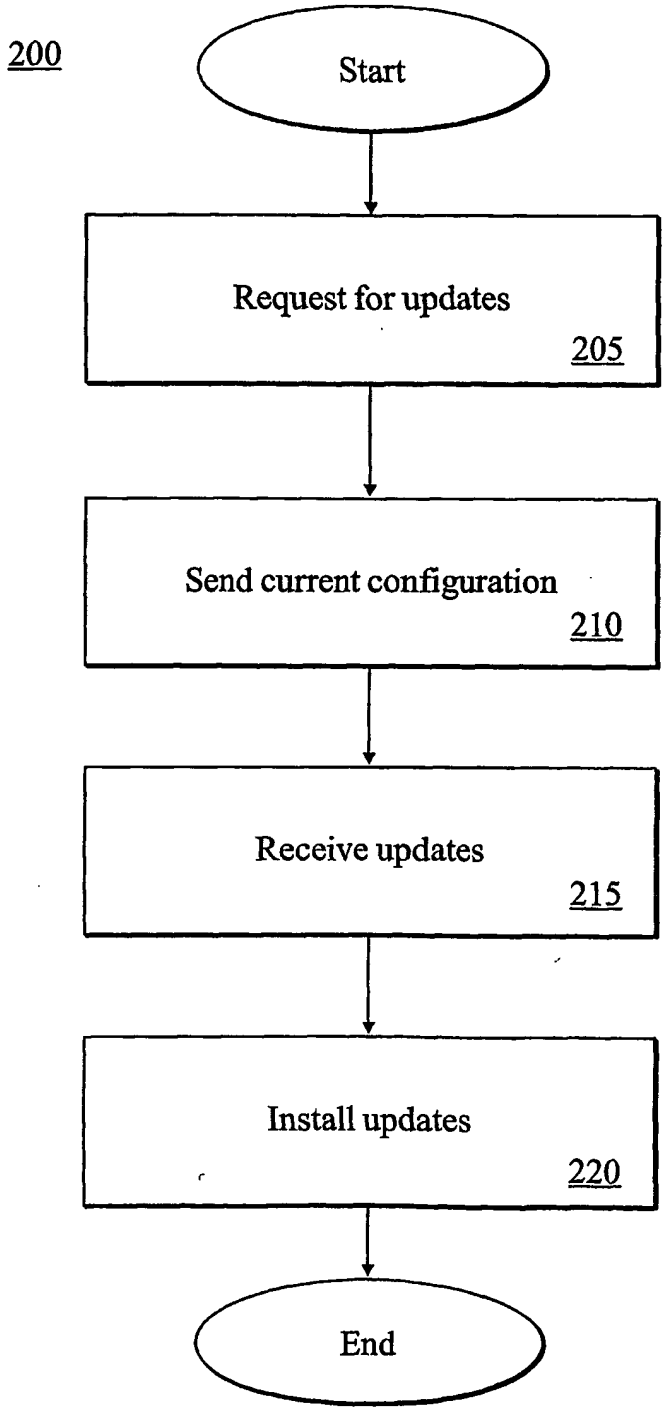

200
Start
Request for updates
205
Send current configuration
210
Receive updates
215
Install updates
220
End

Fig. 2

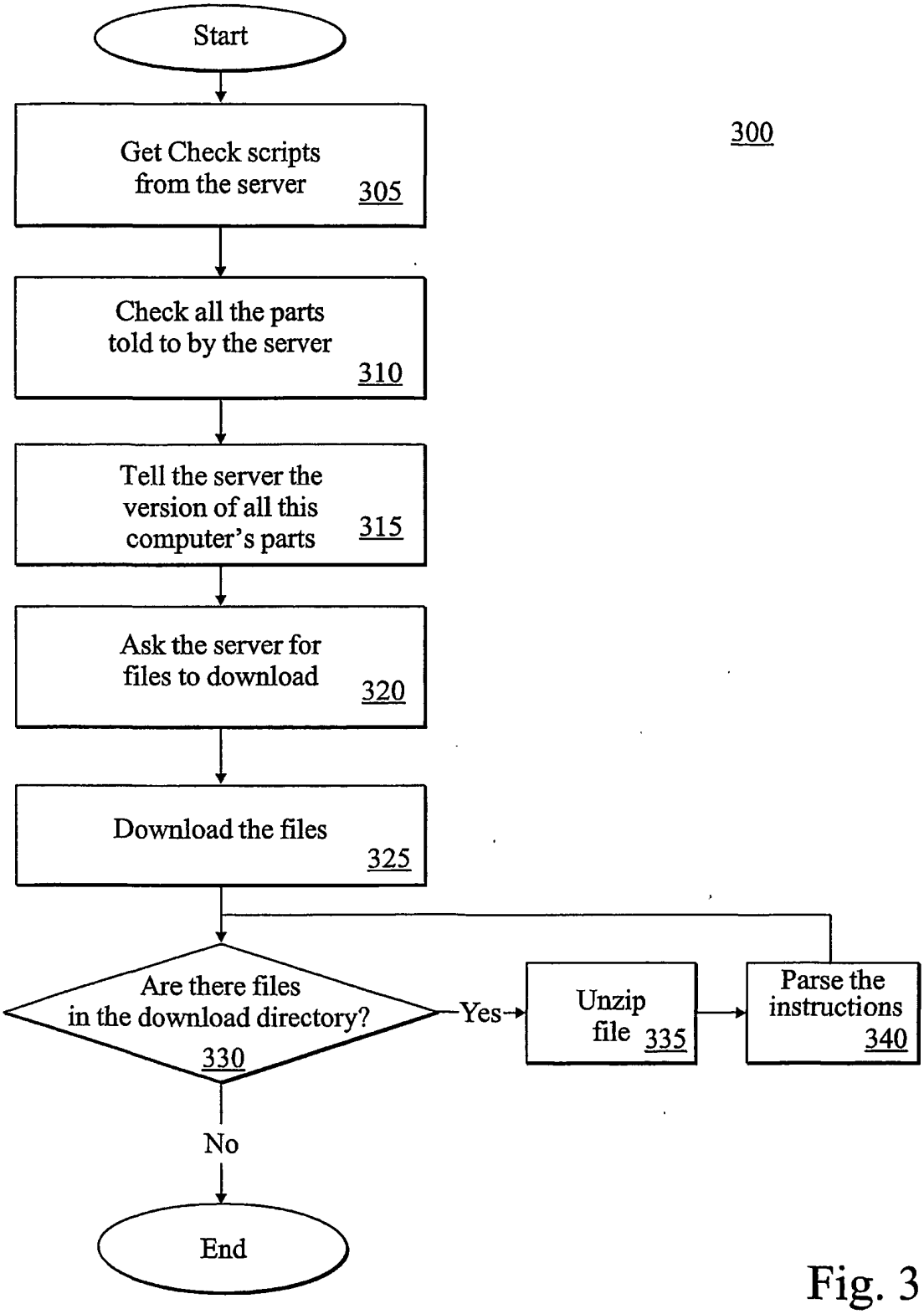

Start
Get Check scripts from the server 305
300
Check all the parts told to by the server 310
Tell the server the version of all this computer's parts 315
Ask the server for files to download 320
Download the files 325
Are there files in the download directory? 330
Yes
Unzip file 335
Parse the instructions 340
No
End

Fig. 3

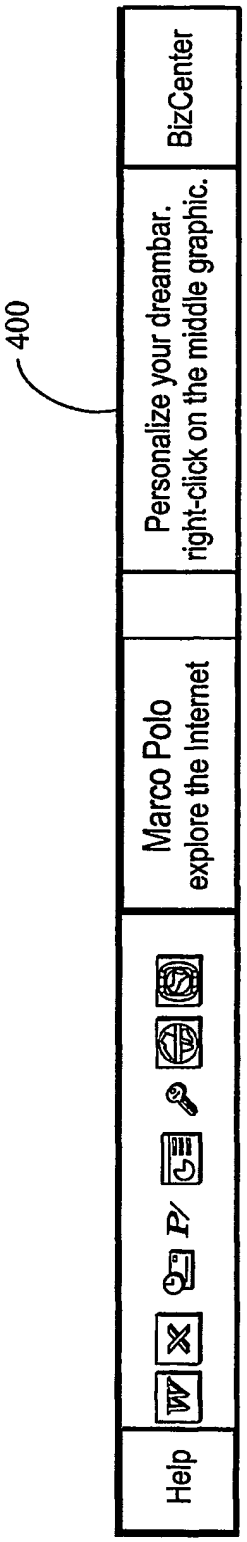

400
Help
Marco Polo
explore the Internet
Personalize your dreambar.
right-click on the middle graphic.
BizCenter

Fig. 4

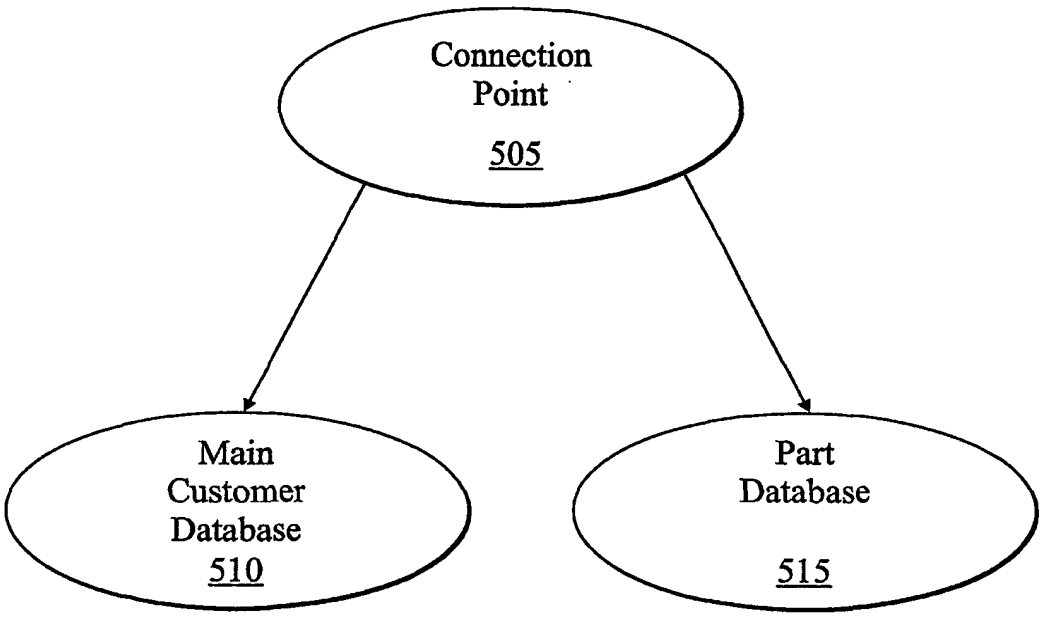

Connection Point
505
Main Customer Database
510
Part Database
515

Fig. 5

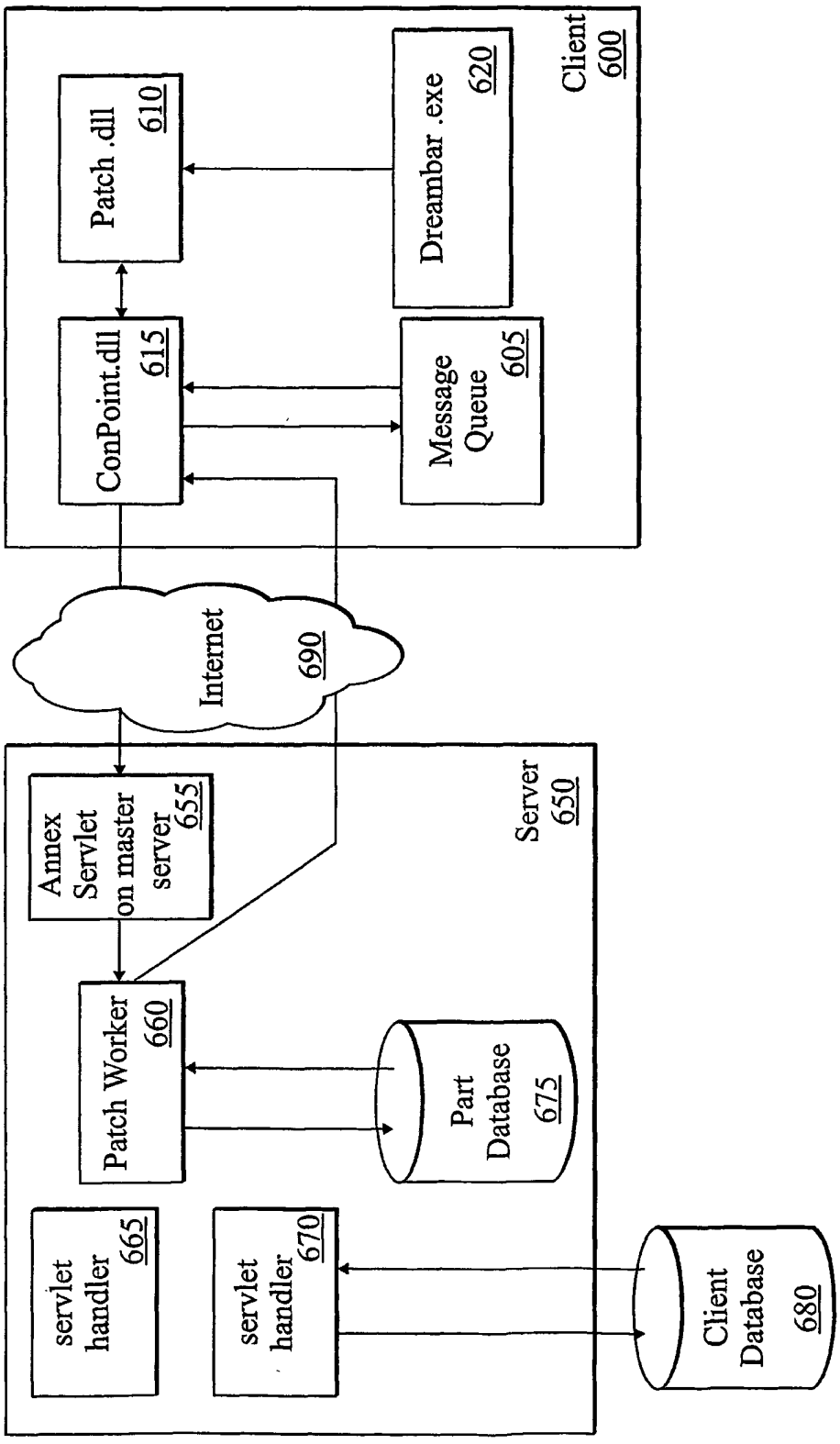

Client 600
Patch .dll 610
Dreambar .exe 620
ConPoint.dll 615
Message Queue 605
Internet 690
Server 650
Annex Servlet on master server 655
Patch Worker 660
Part Database 675
servlet handler 665
servlet handler 670
Client Database 680

Fig. 6

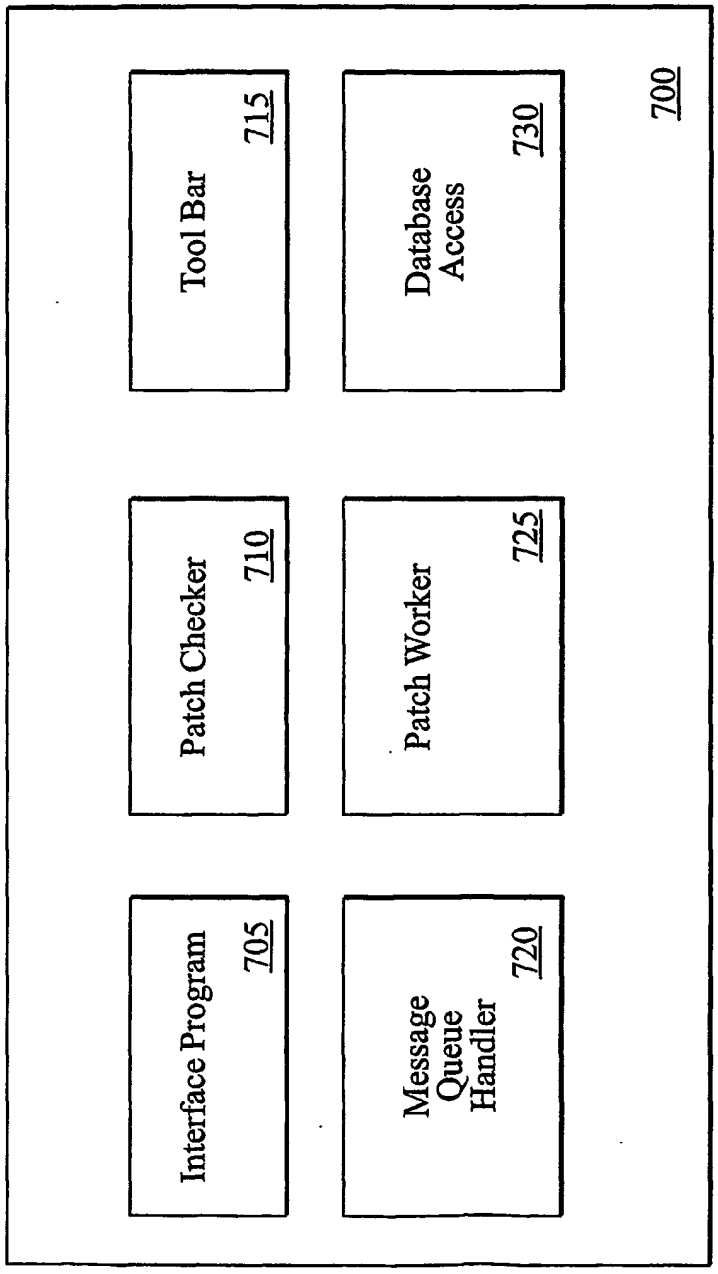

700
Tool Bar
715
Database Access
730
Patch Checker
710
Patch Worker
725
Interface Program
705
Message Queue Handler
720

Fig. 7

INTERNATIONAL SEARCH REPORT

Internation Ication No
PCT/US 01/15720

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/445 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 845 077 A (FAWCETT PHILIP E) 1 December 1998 (1998-12-01) column 4, line 60 -column 9, line 27 ---	1-59
A	EP 0 809 182 A (NIPPON ELECTRIC CO) 26 November 1997 (1997-11-26) column 4, line 3 -column 5, line 53 ---	1,2,19, 20,37, 38,41, 42,46,47
A	US 5 752 042 A (PRITKO STEVEN MICHAEL ET AL) 12 May 1998 (1998-05-12) column 3, line 7 -column 6, line 60 ---	1-59
A	US 6 006 034 A (PORT GRAEME ET AL) 21 December 1999 (1999-12-21) column 4, line 59 -column 6, last line -----	1-59

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 October 2001	06/11/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bijn, K

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Internation cation No
PCT/US 01/15720

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5845077 A	01-12-1998	US 6073214 A	06-06-2000
EP 0809182 A	26-11-1997	JP 9305675 A	28-11-1997
		AU 2348897 A	27-11-1997
		CA 2204317 A1	20-11-1997
		EP 0809182 A1	26-11-1997
US 5752042 A	12-05-1998	US 6074434 A	13-06-2000
US 6006034 A	21-12-1999	NONE	